TD 3 : Les boucles

Les boucles sont basées sur le principe de récurrence :

- On suppose connu l'état d'un ensemble de variables et de propriétés de ces variables à une étape n de la boucle, et une itération de la boucle consiste à calculer les nouvelles valeurs à l'étape n+1 en conservant ces propriétés.
- Si les variables sont correctement initialisées et les propriétés vérifiées avant l'étape 1, alors on peut calculer n'importe quelle étape n en démarrant le calcul à l'étape 1.

Le contrôle de la boucle se basera sur un test qui décidera de l'arrêt sur une configuration particulière des variables modifiables par cette boucle.

1) Suite de cardinal connu et inconnu

Ecrire le programme qui lit une suite de notes au clavier et calcule la moyenne de ces notes. On supposera dans un premier temps que l'on connaît la taille de la suite (N), puis dans un deuxième temps que cette suite est de taille inconnue mais se termine par un nombre négatif.

2) Un peu d'algo autour des boucles

Enrichir le programme précédent pour qu'il trouve aussi la meilleure note dans la suite, ainsi que le nombre de fois où cette meilleure note a été attribuée.

3) La récurrence

Ecrire le programme qui détermine la $n^{ème}$ valeur de la suite de Fibonacci définie comme suit :
$u_1 = 1,\ u_2 = 1,\ u_n = u_{n-1} + u_{n-2}$ pour $n > 2$

4) Trouver le « zéro » d'une fonction croissante par dichotomie

Ecrire le programme qui approche à epsilon près la racine carrée d'un réel positif x connu.
Pour cela, on cherche à trouver le zéro de la fonction $f(r) = r^2 - x$
On fera les constatations suivantes :
$\sqrt{x}$ est compris entre 0 et max(1, x)
Supposons $\sqrt{x} \in [a, b]$, soit $m = (a+b)/2$, si $x - m^2 < 0$ alors $\sqrt{x} \in [a, m]$, sinon $\sqrt{x} \in [m, b]$
En un test, on réduit de moitié l'intervalle contenant la solution. On peut donc réduire l'intervalle [a, b] en conservant la propriété $\sqrt{x} \in [a, b]$.
En relançant l'opération plusieurs fois, on fait converger a et b vers $\sqrt{x}$
Si $\sqrt{x} \in [a, b]$ et $(b-a) <$ epsilon, alors $(a+b)/2$ est une solution approchée satisfaisante.

Correction TD 3 : Les boucles

Suite de cardinal connu et inconnu

Ecrire le programme qui lit une suite de notes au clavier et calcule la moyenne de ces notes. On supposera que cette suite est de taille inconnue mais se termine par un nombre négatif.

Un peu d'algo autour des boucles

Enrichir le programme précédent pour qu'il trouve aussi la meilleure note dans la suite, ainsi que le nombre de fois ou cette moins bonne note a été attribuée.

```
/* Moyenne et plus grande valeur d'une suite */
/* auteur : Menvussa Gerard */
/* date : 30/2/1903 */
#include <stdio.h>

void main(void)
{ int compteur, nbMax;        /* Commentaires facultatifs  */
  float note, max, moyenne ;  /* pour des var bien nommées */
  printf("donnez une suite de notes terminee par -1\n");
  scanf("%f", &note);
  compteur = 1;
  max = moyenne = note;
  nbMax = 1;
  scanf("%f", &note);
  while(note >= 0)
  { /* traitement de la note courante */
    compteur++;
    moyenne = moyenne+note;
    if (max == note) nbMax++;
    else if (note > max)
    { nbMax = 1;
      max = note;
    }
    /* lecture de la note suivante */
    scanf("%f", &note);
  }
  moyenne = moyenne/compteur;
  printf("la moyenne est de %f\n", moyenne) ;
  printf("il y a %d fois la note max %f\n", nbMax, max);
}
```

La récurrence

Ecrire le programme qui détermine la $n^{ème}$ valeur de la suite de Fibonacci définie comme suit :

$u_1 = 1,\ u_2 = 1,\ u_n = u_{n-1} + u_{n-2}$ pour $n > 2$

```
/* Suite de Fibonacci */
/* auteur : Ducable Jean-Raoul */
/* date : 30/2/1903 */
#include <stdio.h>

void main(void)
{ /* élément courant */
  int Un
  /* éléments précédents */
  int Un_1, Un_2 ;
  /* autres variables */
  int n, i ;
  printf("n° de l'element de la suite de Fibonacci ?\n");
  scanf("%d", &n);
  Un_1 = Un = 1;
  for (i=3 ; i<=n ; i++)
  { Un_2 = Un_1;
    Un_1 = Un;
    Un = Un_1+Un_2;
    /* trace à l'ecran*/
    printf("U%d = %d\n", i, Un);
  }
  printf("\nU%d = %d\n", n, Un);
}
```

Trouver le « zéro » d'une fonction croissante.

```
/* Approximation par dichotomie de la racine carrée de x */
/* auteur : Pons Pierre */
/* date : c'est noyel */
#include <stdio.h>
#define EPSILON 0.00001

void main(void)
{   float min, max, m ; /* bornes et milieu de l'intervalle */
    float x; /* valeur dont on cherche la racine */
    printf("donnez un nombre\n");
    scanf ("%f", &x) ;
    /* première délimitation de la solution */
    min = 0 ;
    if (x<1)
      max = 1;
    else
      max = x;
    /*réduction de l'intervalle jusqu'à satisfaire l'erreur*/
    do
    { /* la solution est dans [min, max] */
      m = (min+max)/2;
      if (m*m > x)
        max = m;
```

```
        else
          min = m;
        /* l'écart max-min a diminue de moitié */
        /* la solution est dans [min, max] */
      }
      while (max-min > EPSILON);
      /* la solution est dans [min, max] */
      m = (min+max)/2 ; /* valeur approchee */
      printf("%f est la racine carree de %f\n", m, x);
}
```